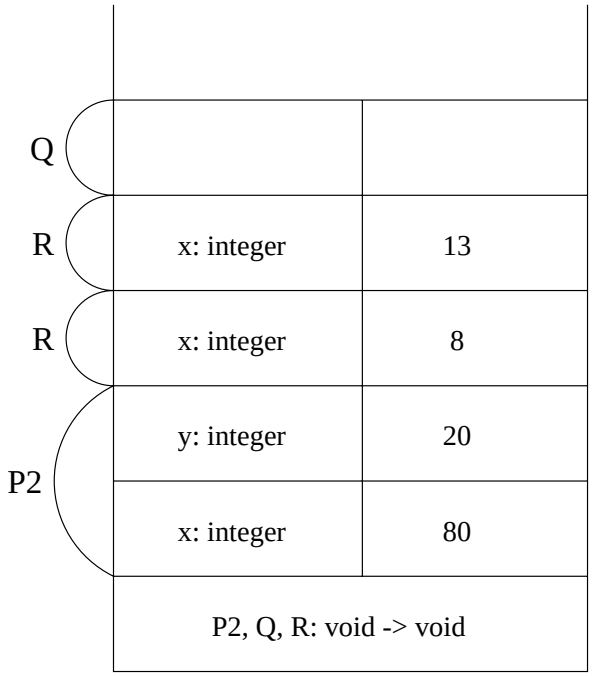
CMSC 330 Exam 2: April 24, 2000

Show all work necessary to justify your answers!

1. (30 points total)

   (a) (10 points)
   If parameters are passed by value, and using static scoping rules, what does the environment (name/value pairs) look like just before procedure Q returns, for the following program, which uses Pascal syntax?

```
program P2;
  var x, y: integer;

  procedure Q;
  begin
    x := y * 4;
  end;

  procedure R;
    var x: integer;
  begin
    x := y + 3;
    y := 2 * y;
    if (x < 10) R
    else Q;
    write(x);
  end;

begin
  x := 2;
  y := 5;
  R;
  writeln(x, y);
end.
```



(b) (20 points)
What does the program output with:

- static scoping,
- dynamic scoping

for non-local variables.
static : 13 8 80 20
dynamic: 80 8 2 20

2. (30 points)
What will the following program, in C++ syntax, output if parameters are passed with:

- call-by-value
- call-by-value-result, with binding of the result at call time
- call-by-value-result, with binding of the result at return time, from left to right in the order of the parameters
- call-by-reference
- call-by-name

```
#include <iostream.h>
int i, j;

int main(int argc, char* argv[])
{
  int k, A[5];

  for (k = 0; k < 5; k++) {
    A[k] = k;
  }
  i = 4;   j = 1;
  g(i, j, A[i], A[j]);
  cout << i << " " << j;
  for (k = 0; k < 5; k++) {
    cout << " " << A[k];
  }
  cout << '\n';
}

void g(int j, int k, int l, int m)
{
  j -= 3;   k += 1;
  l *= 3;   m *= 2;
  i += k;
}
```

value: 6 1 0 1 2 3 4

value-result (bind at call): 1 2 0 2 2 3 12

value-result (bind at return): 1 2 0 12 2 3 4

reference: 3 2 0 2 2 3 12

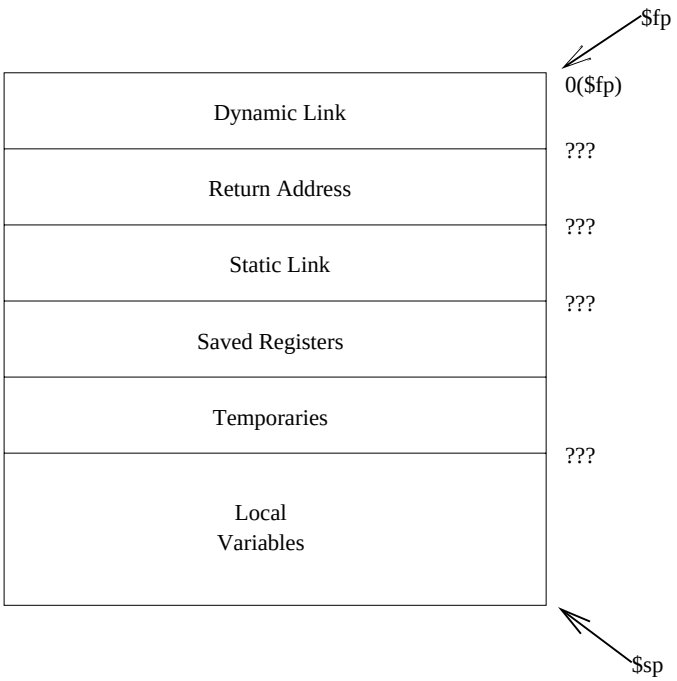
call-by-name: 3 2 0 3 4 3 4

3. (40 points total)
This question requires you to write pseudo-code for several MIPS assembly language instructions that must be executed to run the following program. Your pseudo-code should be as detailed as the assembly instructions, but we are not concerned about syntax. For example, the instruction "sw $fp, 0($sp)" can be written as "store the frame pointer at 0 past the stack pointer."

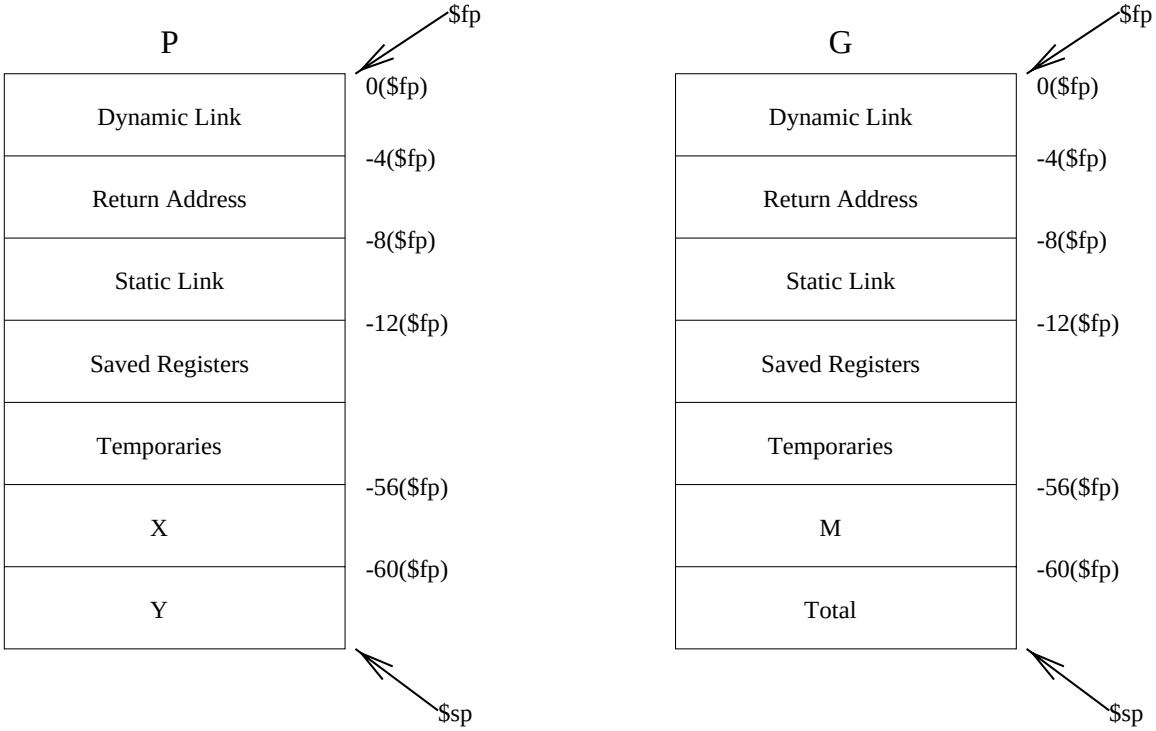
```
program P;
  var X, Y: integer;

  procedure G(M: integer, var Total: integer);

    procedure Print(Val: integer);
    begin
      writeln(Val, Total);
    end;

  begin
    if (M >= 1) then
    begin
      M := M - 1;
      Total := Total + M + 1;
      G(M, Total)
    end
    else
      Print(Total)
  end;

begin
  X := 8;
  Y := 1;
  G(X, Y);
end.
```

You should assume the register conventions discussed in class, and parameter passing is done as was shown in class for Pascal (i.e. default is pass-by-value, unless explicitly specified pass-by-reference in a procedure header with `var`).

The template for a single stack frame (activation record instance) is shown below.



(a) (10 points)
Show a complete layout for the stack frames (activation record instances) for procedures `P` and `G`, based on the template. Fill in all the missing relative addresses (marked by ??? in the template), and make sure to show all local variables, and their locations in the stack frame relative to the frame pointer. Remember that, by convention, the saved registers and temporaries use 44 bytes.



(b) (25 points)
Provide the pseudo-code that is necessary to set up the runtime stack when procedure G calls itself recursively, for both the caller and the callee. Use the stack frame layout for G from the first part of this question to obtain the addresses of local variables.

In caller:

```
lw $a0, -8($fp)        # static link (same as static link of current
                       # activation of G)
lw $a1, -56($fp)       # get M as first parameter
lw $a2, -60($fp)       # get Total as second parameter
jal G
```

In callee:

```
sw $fp, 0($sp)         # store dynamic link (old frame pointer)
sw $ra, -4($sp)        # save return address (for recursive calls)
sw $a0, -8($sp)        # save static link
sw $a1, -56($sp)       # store parameter M
sw $a2, -60($sp)       # store parameter Total
move $fp, $sp          # set new frame pointer
sub $sp, $sp, 64       # set new stack pointer
```

(c) (5 points)
What does the program print?

37 37